**logcat Command-line Tool**

**In this document**

1. [Command-line Syntax](https://developer.android.com/studio/command-line/logcat.html#Syntax)
2. [Starting LogCat](https://developer.android.com/studio/command-line/logcat.html#startingLogcat)
3. [Filtering Log Output](https://developer.android.com/studio/command-line/logcat.html#filteringOutput)
4. [Controlling Log Output Format](https://developer.android.com/studio/command-line/logcat.html#outputFormat)
5. [Viewing Alternative Log Output Buffers](https://developer.android.com/studio/command-line/logcat.html#alternativeBuffers)
6. [Viewing stdout and stderr](https://developer.android.com/studio/command-line/logcat.html#viewingStd)
7. [Logging from Code](https://developer.android.com/studio/command-line/logcat.html#logClass)

Logcat is a command-line tool that dumps a log of system messages, including stack traces when the device throws an error and messages that you have written from your app with the [Log](https://developer.android.com/reference/android/util/Log.html) class.

This page is about the command-line logcat tool, but you can also view log messages from the **Logcat** window in Android Studio. For information about viewing and filtering logs from Android Studio, see [Write and View Logs with Logcat](https://developer.android.com/studio/debug/am-logcat.html).

**Command-line Syntax**

[adb] logcat [<option>] ... [<filter-spec>] ...

You can run logcat as an adb command or directly in a shell prompt of your emulator or connected device. To view log output using adb, navigate to your SDK platform-tools/ directory and execute:

$ adb logcat

You can create a shell connection to a device and execute:

$ adb shell  
# logcat

**Options**

The following table describes the command line options of logcat.

|  |  |
| --- | --- |
| **Option** | **Description** |
| -b <buffer> | Loads an alternate log buffer for viewing, such as events or radio. The main buffer is used by default. See [Viewing Alternative Log Buffers](https://developer.android.com/tools/debugging/debugging-log.html#alternativeBuffers). |
| -c | Clears (flushes) the entire log and exits. |
| -d | Dumps the log to the screen and exits. |
| -f <filename> | Writes log message output to <filename>. The default is stdout. |
| -g | Prints the size of the specified log buffer and exits. |
| -n <count> | Sets the maximum number of rotated logs to <count>. The default value is 4. Requires the -r option. |
| -r <kbytes> | Rotates the log file every <kbytes> of output. The default value is 16. Requires the -f option. |
| -s | Sets the default filter spec to silent. |
| -v <format> | Sets the output format for log messages. The default is brief format. For a list of supported formats, see [Controlling Log Output Format](https://developer.android.com/tools/debugging/debugging-log.html#outputFormat). |

**Starting logcat**

To run logcat, through the ADB shell, the general usage is:

[adb] logcat [<option>] ... [<filter-spec>] ...

You can use the logcat command from your development computer or from a remote adb shell in an emulator/device instance. To view log output in your development computer, you use

$ adb logcat

and from a remote adb shell you use

# logcat

The following table describes the logcat command line options:

|  |  |
| --- | --- |
| -c | Clears (flushes) the entire log and exits. |
| -d | Dumps the log to the screen and exits. |
| -f <filename> | Writes log message output to <filename>. The default is stdout. |
| -g | Prints the size of the specified log buffer and exits. |
| -n <count> | Sets the maximum number of rotated logs to <count>. The default value is 4. Requires the -r option. |
| -r <kbytes> | Rotates the log file every <kbytes> of output. The default value is 16. Requires the -f option. |
| -s | Sets the default filter spec to silent. |
| -v <format> | Sets the output format for log messages. The default is brief format. For a list of supported formats, see [Controlling Log Output Format](https://developer.android.com/studio/command-line/logcat.html#outputFormat). |

**Filtering Log Output**

Every Android log message has a *tag* and a *priority* associated with it.

* The tag of a log message is a short string indicating the system component from which the message originates (for example, "View" for the view system).
* The priority is one of the following character values, ordered from lowest to highest priority:
  + V — Verbose (lowest priority)
  + D — Debug
  + I — Info
  + W — Warning
  + E — Error
  + F — Fatal
  + S — Silent (highest priority, on which nothing is ever printed)

You can obtain a list of tags used in the system, together with priorities, by running logcat and observing the first two columns of each message, given as <priority>/<tag>.

Here's an example of logcat output that shows that the message relates to priority level "I" and tag "ActivityManager":

I/ActivityManager(  585): Starting activity: Intent { action=android.intent.action...}

To reduce the log output to a manageable level, you can restrict log output using *filter expressions*. Filter expressions let you indicate to the system the tags-priority combinations that you are interested in — the system suppresses other messages for the specified tags.

A filter expression follows this format tag:priority ..., where tag indicates the tag of interest and priority indicates the *minimum* level of priority to report for that tag. Messages for that tag at or above the specified priority are written to the log. You can supply any number of tag:priority specifications in a single filter expression. The series of specifications is whitespace-delimited.

Here's an example of a filter expression that suppresses all log messages except those with the tag "ActivityManager", at priority "Info" or above, and all log messages with tag "MyApp", with priority "Debug" or above:

adb logcat ActivityManager:I MyApp:D \*:S

The final element in the above expression, \*:S, sets the priority level for all tags to "silent", thus ensuring only log messages with "ActivityManager" and "MyApp" are displayed. Using \*:S is an excellent way to ensure that log output is restricted to the filters that you have explicitly specified — it lets your filters serve as a "whitelist" for log output.

The following filter expression displays all log messages with priority level "warning" and higher, on all tags:

adb logcat \*:W

If you're running logcat from your development computer (versus running it on a remote adb shell), you can also set a default filter expression by exporting a value for the environment variable ANDROID\_LOG\_TAGS:

export ANDROID\_LOG\_TAGS="ActivityManager:I MyApp:D \*:S"

Note that ANDROID\_LOG\_TAGS filter is not exported to the emulator/device instance, if you are running logcat from a remote shell or using adb shell logcat.

**Controlling Log Output Format**

Log messages contain a number of metadata fields, in addition to the tag and priority. You can modify the output format for messages so that they display a specific metadata field. To do so, you use the -v option and specify one of the supported output formats listed below.

* brief — Display priority/tag and PID of the process issuing the message (the default format).
* process — Display PID only.
* tag — Display the priority/tag only.
* raw — Display the raw log message, with no other metadata fields.
* time — Display the date, invocation time, priority/tag, and PID of the process issuing the message.
* threadtime — Display the date, invocation time, priority, tag, and the PID and TID of the thread issuing the message.
* long — Display all metadata fields and separate messages with blank lines.

When starting logcat, you can specify the output format you want by using the -v option:

[adb] logcat [-v <format>]

Here's an example that shows how to generate messages in thread output format:

adb logcat -v thread

Note that you can only specify one output format with the -v option.

**Viewing Alternative Log Buffers**

The Android logging system keeps multiple circular buffers for log messages, and not all of the log messages are sent to the default circular buffer. To see additional log messages, you can run the logcat command with the -b option, to request viewing of an alternate circular buffer. You can view any of these alternate buffers:

* radio — View the buffer that contains radio/telephony related messages.
* events — View the buffer containing events-related messages.
* main — View the main log buffer (default)

The usage of the -b option is:

[adb] logcat [-b <buffer>]

Here's an example of how to view a log buffer containing radio and telephony messages:

adb logcat -b radio

**Viewing stdout and stderr**

By default, the Android system sends stdout and stderr (System.out and System.err) output to /dev/null. In processes that run the Dalvik VM, you can have the system write a copy of the output to the log file. In this case, the system writes the messages to the log using the log tags stdout and stderr, both with priority I.

To route the output in this way, you stop a running emulator/device instance and then use the shell command setprop to enable the redirection of output. Here's how you do it:

$ adb shell stop  
$ adb shell setprop log.redirect-stdio true  
$ adb shell start

The system retains this setting until you terminate the emulator/device instance. To use the setting as a default on the emulator/device instance, you can add an entry to /data/local.prop on the device.

**Logging from Code**

The [Log](https://developer.android.com/reference/android/util/Log.html) class allows you to create log entries in your code that display in the logcat tool. Common logging methods include:

* [Log.v(String, String)](https://developer.android.com/reference/android/util/Log.html#v%28java.lang.String,%20java.lang.String%29) (verbose)
* [Log.d(String, String)](https://developer.android.com/reference/android/util/Log.html#d%28java.lang.String,%20java.lang.String%29) (debug)
* [Log.i(String, String)](https://developer.android.com/reference/android/util/Log.html#i%28java.lang.String,%20java.lang.String%29) (information)
* [Log.w(String, String)](https://developer.android.com/reference/android/util/Log.html#w%28java.lang.String,%20java.lang.String%29) (warning)
* [Log.e(String, String)](https://developer.android.com/reference/android/util/Log.html#e%28java.lang.String,%20java.lang.String%29) (error)

For example, using the following call:

Log.i("MyActivity", "MyClass.getView() — get item number " + position);

The logcat outputs something like:

I/MyActivity( 1557): MyClass.getView() — get item number 1